

REMARKS:Status

Claims 1 to 29 are pending. The independent claims, namely claims 1, 7, 24 and 27, are amended herein. Entry of this amendment, reconsideration and further examination are respectfully requested.

Response to Arguments

Two main points were made in the Response to Arguments section of the Office Action. The first point was that the “features upon which applicant relies (i.e., single statically-allocated thread) are not recited in the claims” (emphasis in original). In reply, Applicant respectfully points out that prior to amendment claim 1 recited “simulating a plurality of dynamically-allocated threads using *a* statically-allocated thread” (emphasis added). While the word “single” did not appear in the claim, the language clearly stated that *a* statically-allocated thread was used to simulate plural dynamically-allocated threads, thus indicating that many threads were simulated by *a* (i.e., a single) statically-allocated thread. Similar language was present in claims 7, 24 and 27 prior to amendment.

Nonetheless, Applicant has now amended claims 1, 7, 24 and 27 to explicitly recite “a *single* statically-allocated thread” (emphasis added). In this regard, Applicant notes that this recitation does not preclude use of other threads in addition to the single statically-allocated thread that simulates the plurality of dynamically-allocated threads. For example, a computer

that implemented the method of claim 1 could run one or more other statically-allocated or dynamically-allocated threads, for example to execute other functions or to simulate plural additional dynamically-allocated threads, without departing from the language of claims 1 and 7.

The Examiner's second point in the Response to Arguments section was disagreement with Applicant's assertion that "in Guedalia, the dynamically-allocated threads are not simulated thread[s]." The Examiner cited col. 3, line 61 et seq. as showing that Guedalia's threads are simulated threads. The cited portion is reproduced below for the sake of convenience:

Modern operating systems support multi-tasking, which is the ability to run many separate applications at the same time. A single software program can take advantage of multi-tasking by creating multiple concurrent "threads." *Each thread simulates a separate application.* Thus an HTTP server, for example, can use multiple threads to optimize its performance in responding to concurrent requests. Each request can be processed in a separate thread, and while one request is being processed in the CPU, a second request can be transmitted through the network hardware. If only one thread is used, then although the network hardware is buffered, the processing of the second request can become blocked--because the single thread waits for the network to finish sending.

(Emphasis added.)

Applicant notes that this portion of Guedalia appears in the Background of the Invention. The text appears to Applicant to be nothing more than a general discussion of a multi-tasking operating system. While the discussion notes that "[e]ach thread simulates a separate *application*," no mention is made of simulating *dynamically-allocated threads*. Instead, Guedalia's Background of the Invention appears to Applicant to be discussing conventional actual threads implemented by a conventional modern operating system. In any event, no

mention appears to Applicant to be made of simulating these threads using a single statically-allocated thread.

### Claim Rejections

Claims 1 to 29 were rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,535,878 (Guedalia). Applicant respectfully traverses this rejection.

Claims 1 to 6 and 12 to 17: Claim 1 is reproduced below:

1. A method including  
simulating a plurality of dynamically-allocated threads using a  
single statically-allocated thread; and  
maintaining state information regarding each dynamically-allocated  
thread maintained within said statically-allocated thread.

The applied art, namely Guedalia, is not seen by Applicant to disclose or to suggest the foregoing features of claim 1, at least with respect to “simulating a plurality of dynamically-allocated threads using a single statically-allocated thread.”

In this regard, the threads in Guedalia are understood by Applicant to be actual dynamically-allocated threads, not simulated threads. In particular, Guedalia specifically recites that his “present invention dynamically allocates threads.” Guedalia, col. 21, lines 8 and 9. Guedalia’s thread manager 116 “concurrently monitors the currently running threads 118.” Guedalia, col. 21, lines 40 to 42. Likewise, Guedalia’s manage threads step 126 “is constantly monitoring the active threads in the thread pool.” Guedalia, col. 22, lines 15 and 16. To Applicant, this language clearly indicates that the threads are actual concurrent dynamically-allocated threads.

In contrast, the method of claim 1 uses “a single statically-allocated thread” to simulate “a plurality of dynamically-allocated threads.”

Guedalia does discuss simulation, but in the context of a server test in which clients were simulated. See Guedalia, col. 20, lines 44 and 45. Applicant sees no indication that these simulated clients resulted in simulated threads, let alone threads simulated by a single statically-allocated thread as recited by claim 1.

As pointed out in the Response to Arguments section of the Office Action, Guedalia’s Background of the Invention notes that “[m]odern operating systems support multi-tasking ... [in which] a single software program can take advantage of multi-tasking by creating multiple concurrent ‘threads.’” Guedalia goes on to state that “[e]ach thread simulates a separate application.” However, Applicant sees no mention of simulating *dynamically-allocated threads* as opposed to using threads to simulate separate *applications*. Furthermore, no mention appears to Applicant to be made of simulating these threads using a single statically-allocated thread.

Applicant also submits that Guedalia is silent with respect to simulating dynamic allocation using static allocation. In this regard, the terms “static” and “statically” do not even appear in Guedalia. This deficiency of Guedalia is believed to distinguish claim 1 even further from that reference.

In view of the foregoing, reconsideration and withdrawal are respectfully requested of the § 102(e) rejection of claim 1 and its dependent claims. Allowance of these claims also is requested.

Claims 7 to 11 and 19 to 23: Claim 7 is reproduced below:

7. Apparatus including a file server system having a single statically-allocated thread including a plurality of simulated dynamically-allocated threads, said statically-allocated thread including state information regarding each said simulated thread.

The applied art, namely Guedalia, is not seen by Applicant to disclose or to suggest the foregoing features of claim 7, at least with respect to “a single statically-allocated thread including a plurality of simulated dynamically-allocated threads.” Accordingly, reconsideration and withdrawal are respectfully requested of the § 102(e) rejection of claim 7 and its dependent claims. Allowance of these claims also is requested.

Claims 24 to 26: Claim 24 is reproduced below:

24. A method of implementing a plurality of simulated dynamically-allocated threads using a single statically-allocated thread, comprising:

using a scheduler implemented by said single statically-allocated thread to call thread blocks for said plurality of simulated dynamically-allocated threads; and

maintaining state information regarding each of said plurality of simulated dynamically-allocated threads.

The applied art, namely Guedalia, is not seen by Applicant to disclose or to suggest the foregoing features of claim 24, at least with respect to “using a scheduler implemented by said single statically-allocated thread to call thread blocks for said plurality of simulated dynamically-allocated threads.” Accordingly, reconsideration and withdrawal are respectfully requested of the § 102(e) rejection of claim 24 and its dependent claims. Allowance of these claims also is requested.

Claims 27 to 29: Claim 27 is reproduced below:

27. Apparatus including a server that implements a plurality of simulated dynamically-allocated threads using a single statically-allocated thread, comprising:

a processor that executes a scheduler implemented by said single statically-allocated thread to call thread blocks for said plurality of simulated dynamically-allocated threads; and

memory that stores state information regarding each of said plurality of simulated dynamically-allocated threads.

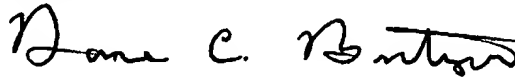
The applied art, namely Guedalia, is not seen by Applicant to disclose or to suggest the foregoing features of claim 27, at least with respect to “a scheduler implemented by said single statically-allocated thread to call thread blocks for said plurality of simulated dynamically-allocated threads.” Accordingly, reconsideration and withdrawal are respectfully requested of the § 102(e) rejection of claim 27 and its dependent claims. Allowance of these claims also is requested.

#### Closing

In view of the foregoing amendments and remarks, the entire application is believed to be in condition for allowance, and such action is respectfully requested at the Examiner's earliest convenience.

Applicant's undersigned attorney can be reached at (614) 486-3585. All correspondence should continue to be directed to the address indicated below.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Dane C. Butzer". The signature is fluid and cursive, with the first name "Dane" being the most prominent.

Dane C. Butzer  
Reg. No. 43,521

Dated: February 4, 2004

The Swernofsky Law Group  
P.O. Box 390013  
Mountain View, CA 94039-0013  
(650) 947-0700